Claims

[c1] *Knowledge-Driven Architecture* is a software architecture comprising:

(a) Knowledgebase containing business rules and application scenarios that reflect application requirements

(b) Application Scenario Player capable of playing scenarios and transforming acts of scenarios and business rules into interactions with knowledgebase, presentation components, and the underlying application services

(c) Service Connector

(d) Presenter components

(e) Service components

(f) Optimizer

(g) Methods enabling:

(i) separation of software in the application layer, which describes interactions and service calls, and the system layer, which describes services

(ii) storage of application layer description that reflects application requirements as business rules and scenarios in the knowledgebase

(iii) interpretation of scenarios and business rules into interactions with knowledgebase, presentation components, and the underlying application services

(iv) creation and modification of business rules and scenarios that comprise the application layer at run-time

(v) invocation of services designed as integration-ready components with differentiated APIs

(vi) translation of snapshots of existing rules and scenarios into source code of a traditional fixed-form application with better performance and less flexibility

(vii) analysis of successful scenario execution including:

a) history of successes

b) history of interpretation failures

c) learning scenarios that prompt an agent (a user or a program) to re-define the input or to provide more details for better interpretation

d) queue of scenarios with un-answered questions to resolve unsuccessful interpretations

(viii) ability to exchange information about knowl-

edge and service elements existing on other distributed network systems built with this architecture

(ix) multiple forms of knowledge and service resource sharing over distributed networks

[c2] Knowledge-driven architecture of claim 1 wherein application scenarios are written with the Application Scenario Language that allows developers or business experts to quickly build the software application layer by describing application flow as a set of scenarios that define interactions between system components and agents, where said agents can be users, human beings, or programs, for example, a partner program engaged in a common business transaction.

[c3] Knowledge-driven architecture of claim 2 wherein application scenarios consist of scenario acts that can define:

(a) interactions between system components and agents

(b) prompt messages to agents, expected agent responses, if any, and rules for interpretation of agent responses

(c) invocations of knowledgebase services, like queries, assertions, etc. using service names and optional arguments

(d) invocations of application services using service and operation names and optional arguments

(e) variables that are replaced with their values at run-time

(f) conditions based on run-time variable values and knowledgebase queries

(g) the order of execution of scenarios and scenario acts

(h) aliases, or multiple ways of expressing the same meaning

(i) translation policies related to input

(j) possible agent events and event handling rules

(k) instructions for presentation components

[c4] Knowledge-driven architecture of claim 1 wherein the Presenter can include

(a) Formatter that prepares data for audio or video interaction or for communication to other programs

(b) Communicator that provides formatted data communications via peer-to-peer distributed networks or any other protocols

(c) Performer that uses formatted data for actual presentation via voice or screen

(d) special engines such as speech, handwriting, or image recognition, which might target a specific type of user input

[c5] Knowledge-driven architecture of claim 1 wherein the knowledgebase component includes a *Knowledge Service* component

[c6] Knowledge-driven architecture of claim 5 wherein the *Knowledge Service* component serves as an adapter to the knowledgebase, adapting the knowledge engine interface to the interface required by the Service Connector

[c7] Knowledge-driven architecture of claim 1 wherein the *Service Connector* component includes:

(a) Object Retrieval that is able to find an existing service object or load the requested service class and instantiate the object at run-time

(b) Object Registry that associates service objects with service and object names, stores service objects, and makes them reusable

(c) Method Retrieval that retrieves the proper service method belonging to a selected service object based on the provided method arguments

(d) Method Performer that performs the requested service operation on the selected service object

[c8] Knowledge-driven architecture of claim 1 wherein the Optimizer takes a snapshot of existing rules and scenarios and translates them into source code in a language

such as Java or C#, which can later be compiled into binary code to fix the current application rules into a regular application that lacks flexibility but provides better performance.

[c9] Knowledge-driven architecture of claim 1 wherein the Scenario Player contains but is not limited to:

(a) Input Type Checker that checks current input and, depending on its type, submits the input to one of several interpreters

(b) One or more interpreters, such as the Scenario Act Interpreter, the Prompt Response Interpreter, the New Agent Request Interpreter, etc. that, based on scenario and knowledgebase rules, translate acts of scenarios or any other input into a direct action by one of the system components

(c) Queue of Scenarios that stores the current scenario when it cannot be executed at the current time, but needs to be executed later

(d) Success Analysis component that can store and retreive a history of interpretation successes and failures, and in the case of interpretation failure invokes one of several learning scenarios that prompt an agent (a user or a program) to re-define the input or to provide more details for a better interpretation

[c10] Knowledge-driven architecture of claim 1 wherein service components are endowed with usage and value properties.

[c11] Knowledge-driven architecture of claim 9 wherein the Success Analysis component maintains and consistently refines a list of previously used services with their APIs, keywords, descriptions, and related scenarios in the knowledgebase, and re-evaluates the usage and value properties of the services.

[c12] Knowledge-driven architecture of claim 9 wherein the New Agent Request interpreter uses the list of previously used services with their APIs, keywords, descriptions, and related scenarios for automatic translation of user requests into service APIs and scenario acts.

[c13] Knowledge-driven architecture of claim 9 wherein the New Agent Request interpreter uses a list of previously used services with their APIs, keywords, descriptions, and related scenarios to offer selected parts of this information to the user for semi-automatic translation of new requests into service APIs and scenario acts.

[c14] Knowledge-driven architecture of claim 1 wherein the Communicator provides collaborative access to knowledge and services existing on other distributed network

systems built with this architecture.

[c15] Knowledge-driven architecture of claim 11 wherein the Success Analysis component propagates via the Communicator to distributed network systems information on a new service API or a new knowledge subject after the first success operation that included the service or the knowledge subject and then after each update provided locally by the Success Analysis component.

[c16] Knowledge-driven architecture of claim 15, wherein information on new elements is propagated after the first successful operation that included the new element, and thereafter after each local update of such information by the Success Analysis component.